

16.11.2015



Test-Driven Development mit dem Sitecore Abstraktions- Framework

Über mich...

Name	Marc Kurz
Kontakt	marc.kurz@ecx.io
Aktuelle Position	Senior Developer & Technical Architect ecx.io austria GmbH
Aufgabenbereich	Software-Qualität, Development, Projektentwicklung, ...



- 01** Einleitung
- 02** Test Driven Development
- 03** Sitecore Abstraktion
- 04** Werkzeuge
- 05** Zusammenfassung & Ausblick

Einleitung

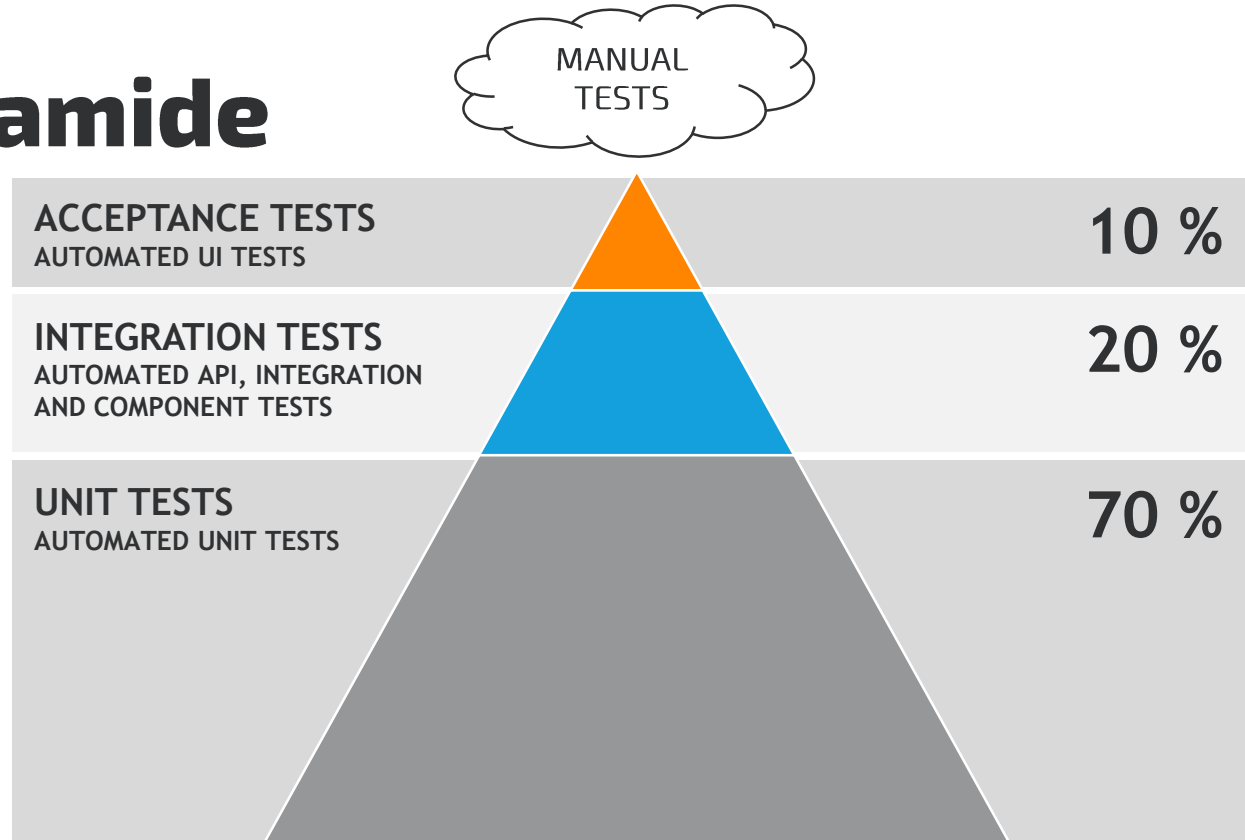
Kundenprojekt

- Code Refactoring der gesamten Solution
- Hohe Testabdeckung des Custom Code als Ziel
 - Erste Ansätze mit „schwergewichtigem“ Mocking Framework: MS Fakes
 - Sehr komplex, schwer einsetzbar
- Generelle Vorgehensweise nach TDD gewünscht
- Problem:
 - Direkte Abhängigkeit zu Sitecore (und anderen Libs)
 - Ausführung benötigt laufende Sitecore-Instanz....

Einleitung

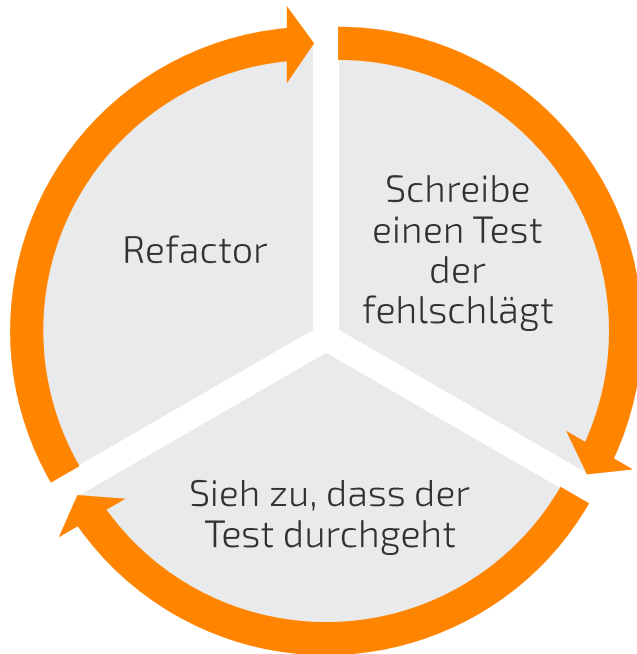
- Implementierung der Solution nach Sitecore Standard
 - Unit Tests sind generell schwierig da das System nicht dahingehend entworfen wurde (bspw. Middlelayer, Abstraktionen, SoC, etc.)
 - Geringe Ausrichtung nach „Design for Testability“
- Unit Tests müssen unabhängig von Sitecore durchgeführt werden
 - Entwicklung von Unit Tests nur unter massivem Einsatz von Fake-Frameworks (bspw. MS Fakes) möglich
- Test-Driven Development Vorgehensweise.....

Testpyramide



- 01** Einleitung
- 02** Test Driven Development
- 03** Sitecore Abstraktion
- 04** Werkzeuge
- 05** Zusammenfassung & Ausblick

Test-Driven Development



Test-Driven Development

Die drei TDD Grundregeln [1]

„You may not write production code until you have written a failing unit test.“

“You may not write more of a unit test than is sufficient to fail, and not compiling is failing.“

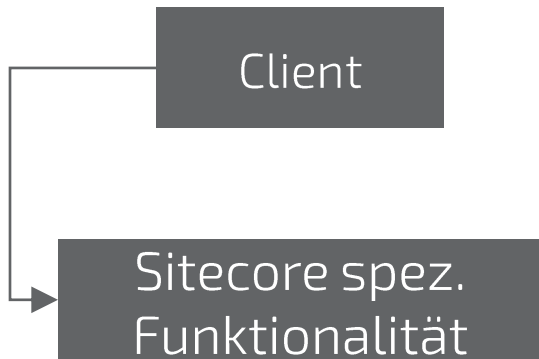
„You may not write more production code than is sufficient to pass the currently failing test“

[1] Martin, Robert C. *Clean code: a handbook of agile software craftsmanship*. Pearson Education, 2009

- 01** Einleitung
- 02** Test Driven Development
- 03** Sitecore Abstraktion
- 04** Werkzeuge
- 05** Zusammenfassung & Ausblick

Sitecore Implementierung

- Implementierung nach Sitecore Standard
- Generell „evolutionäre“ Architektur



Bspw. `Sitecore.Context.Database.GetItem()`

```
/// <summary>
/// Gets filter
/// </summary>
/// <param name="item">Given item</param>
/// <returns>List of items</returns>
public static List<Item> GetFilter(Item item)
{
    List<Field> fields = item.Fields.Where(x => x.Section.Equals("Sho
    List<Item> list = new List<Item>();
    foreach (Field field in fields)
    {
        if (!string.IsNullOrEmpty(field.Value) && field.Type.Equals("
        {
            Item item = Sitecore.Context.Database.GetItem(field.Value
            if (item != null)
            {
                list.Add(item);
            }
        }
    }
    return list;
}
```

Dependency Inversion

High-level modules should not depend on low-level modules.

Both should depend on abstractions [2]

Abstractions should not depend on details. Details should depend upon abstractions [2]

Dependency Inversion

Dependency Injection

- Software pattern, welches IoC implementiert (folgt also dem Dependency Inversion Prinzip)
- Eine "Injection" ist das Übergeben ("injizieren") einer Abhängigkeit zu einem Client

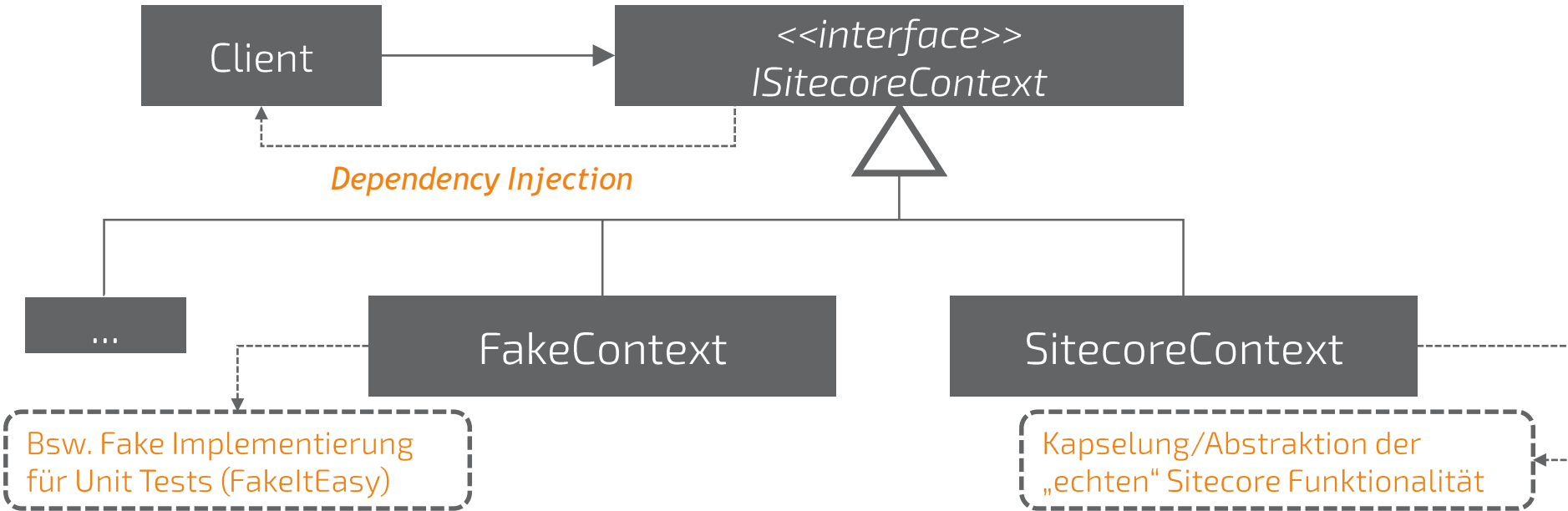
Inversion of Control

- Entkoppelt Implementierung und Ausführung
- Klassen auf höheren Ebenen arbeiten nicht direct mit low-level Klassen, sie benutzen Interfaces als abstrakten Layer


"Don't call us, we'll call you."

The Hollywood Principle, Implicit Invocation & Inversion Of Control

Abstraktionsframework



Abstraktionsframework - Eckdaten

- 115 Interfaces
 - 51 Helper Classes
 - 1027 Tests (~30%)
- 
- Stand
Nov. 2015

Context, Item, ChildList, FieldCollection, Language,
DeviceItem, MediaItem, TemplateItem, ItemManager,
LanguageManager, Template, Log, FileUtil, LinkManager,
UrlOptions, MediaManager, ItemAccess, SecurityDisabler,
Site, SiteManager, MainUtil, StringUtil, ...

Abstraktionsframework - Eckdaten

- 115 Interfaces
- 51 Helper Classes
- 1027 Tests (~30%)

Stand
Nov. 2015

```
public static class Global
{
    Global Configuration Settings (Object Injection)

    #region Initializations

    /// <summary>
    /// The initialize sitecore context.
    /// </summary>
    public static void InitializeSitecoreContext()...

    /// <summary>
    /// The initialize fake context.
    /// </summary>
    public static void InitializeFakeContext()...

    #endregion
}
```

- Initialisierung in EventsAndPipelines.config

```
<processor patch:before="*[@type='Sitecore.Pipelines.Loader.ShowVersion, Sitecore.Kernel']"
type="
    Application.Pipelines.Initialize.InitializeSitecoreContext,
"></processor>
```


- 01** Einleitung
- 02** Test Driven Development
- 03** Sitecore Abstraktion
- 04** Werkzeuge
- 05** Zusammenfassung & Ausblick

Werkzeuge – Specflows (Cucumber)

- Specflows (Cucumber for .NET)
 - Define Tests in a Given-When-Then Form
 - Definierbar bspw. in Jira

Feature: Image10r15ColPresenter_Test

Scenario Outline: Image10r15ColPresenter with different inputs

Given an initialized Sitecore Context

And we use these files for initialize the items:

FileName
2015-07-30_Image10r15ColPresenter_Items.xml

When the ImageColPresenter is called with '<datasourceItem>'

Then we expect the presenter ImageColPresenter to deliver '<result>'

Scenarios:

datasourceItem		result
/sitecore/content	test/Copy of Module Parsing Test Area/Image10r15Col/Content/Modules/Copy of Image10r15Col 1	True
/sitecore/content	test/Copy of Module Parsing Test Area/Image10r15Col/Content/Modules/Image10r15Col	False



Methodenstubs (i.e. „Steps“) werden automatisch generiert

Werkzeuge – Specflows (Cucumber)

– Item (abstrahierte Items) Generieren direkt in Specflows

```

● Given an initialized Sitecore Context
● And we use these files for initialize the items:
3 | FileName |
  | 2015-04-22_Templates_WebsiteRelaunch2012.xml |

● And we create new custom fake items in the 'master' database
#Required columns: ItemName, TemplateId, TemplateName, ParentItemPath, Language, ItemId or FullPath
3 | ItemId | ItemName | FullPath | ParentItemPath |
  | {AAAAAAA-42B8-4314-9D2A-345766720BD0} | test-page-11 | | | /sitecore/conten
  | {4498D90C-42B8-4314-9D2A-345766720BD0} | test-page-12 | /sitecore/content | /test-page-1/test-page-12 | /sitecore/conten
  | {4498D90C-42B8-4314-9D2A-345766720BD0} | test-page-1 | /sitecore/content | /test-page-1 | /sitecore/conten
  | | test-page-21 | /sitecore/content | /test-page-2/test-page-21 | /sitecore/conten
  | | test-page-2 | /sitecore/content | /test-page-2 | /sitecore/conten

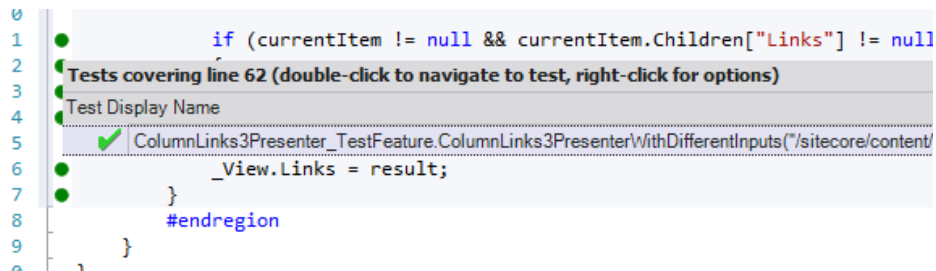
● And we set the field values on one custom fake item. ItemId: '{4498D90C-42B8-4314-9D2A-345766720BD0}' ItemPath: ''
#The field values should be in raw value format
#Required columns: field name, field value
3 | field id | field name | field value |
  | | Navigation Title | Test page 1 |
  | | Hide From Navigation | 1 |
  | {5DD74568-4D4B-44C1-B513-0AF5F4CDA34F} | __Created by | admin |

● And we set the field values on one custom fake item. ItemId: '' ItemPath: '/sitecore/content/'
#The field values should be in raw value format
#Required columns: field name, field value
3 | field id | field name | field value |
  | | Navigation Title | Test page 2 |
  | | Hide From Navigation | 1 |
  | {5DD74568-4D4B-44C1-B513-0AF5F4CDA34F} | __Created by | admin |

```

Werkzeuge – Ncrunch

- Ncrunch (an automated concurrent testing tool for Visual Studio)
 - Features
 - Automatic Concurrent Testing
 - Code Coverage Metrics
 - Performance Metrics



Werkzeuge – FakeItEasy

- FakeItEasy (a .Net dynamic fake framework for creating all types of fake objects, mocks, stubs etc.)

```
// Creating a fake object is just dead easy!  
// No mocks, no stubs, everything's a fake!  
var lollipop = A.Fake<ICandy>();  
var shop = A.Fake<ICandyShop>();  
// Easily set up a call to return a value  
A.CallTo(() => shop.GetTopSellingCandy()).Returns(lollipop);  
// Use your fake as you would an instance of the faked type.  
var developer = new SweetTooth();  
developer.BuyTastiestCandy(shop);  
// Asserting uses the same syntax as configuring calls.  
// There's no need to learn another syntax.  
A.CallTo(() => shop.BuyCandy(lollipop)).MustHaveHappened();
```

Siehe <https://github.com/FakeItEasy/FakeItEasy>

Werkzeuge – FakeItEasy

- FakeItEasy (a .Net dynamic fake framework for creating all types of fake objects, mocks, stubs etc.)

```
/// <summary>
/// Gets or sets the fake inner item.
/// </summary>
/// <value>
/// The fake inner item.
/// </value>
public IItem FakeInnerItem
{
    get
    {
        return this.fakeInnerItem ?? (this.fakeInnerItem = A.Fake<IItem>());
    }

    set
    {
        this.fakeInnerItem = value;
    }
}
```

```
this.fakeItemID = itemID;
this.fakeDatabase = database;
this.itemName = name;
this.parentID = parentID;
this.FakeFields = fields ?? new List<IField>();
this.FakeChildren = children ?? new List<IItem>();
this.fakeLanguage = language;
this.fakeLanguages = languages ?? new[] { language };
this.TemplateID = templateID;
this.templateName = templateName;
this.path = fullPath;
this.version = version;
this.FakeUrl = url;

A.CallTo(this.FakeInnerItem).Where(call => call.Method.Name == "Paths.FullPath")
    .WithReturnType<string>()
    .Returns(this.FakeFullPath);
```

- 01** Einleitung
- 02** Test Driven Development
- 03** Sitecore Abstraktion
- 04** Werkzeuge
- 05** Zusammenfassung & Ausblick

HIGHLIGHTS

x;o

Entkoppelung
von Sitecore

Testability

Test Driven
Development

HIGHLIGHTS

X;O

Entkoppelung
von Sitecore

```
using Sitecore;  
using Sitecore.Data;  
using Sitecore.Layouts;  
using Sitecore.Web.UI.WebControls;
```

```
/// <summary>  
/// Gets filter  
/// </summary>  
/// <param name="item">Given item</param>  
/// <returns>List of items</returns>  
public static List<Item> GetFilter(Item item)  
{  
    List<Field> fields = item.Fields.Where(x => x.Section.Equals("Shop Filter Data"));  
    List<Item> list = new List<Item>();  
    foreach (Field field in fields)  
    {  
        if (!string.IsNullOrEmpty(field.Value) && field.Type.Equals("Text"))  
        {  
            Item item = Sitecore.Context.Database.GetItem(field.Value);  
            if (item != null)  
            {  
                list.Add(item);  
            }  
        }  
    }  
    return list;  
}
```

Testability

```
using Sitecore;  
using Sitecore.Data;  
using Sitecore.Layouts;  
using Sitecore.Web.UI.WebControls;
```

```
/// <summary>  
/// Gets filter  
/// </summary>  
/// <param name="item">Given item</param>  
/// <returns>List of items</returns>  
public static List<IItem> GetFilter(IItem item)  
{  
    var fields = item.Fields.Where(x => x.Section.Equals("Shop Filter Data") && x.Name != "Section");  
    var list = new List<IItem>();  
    foreach (var field in fields.Where(field => !string.IsNullOrEmpty(field.Value) && field.Type.Equals("Text")))  
    {  
        var fieldValueItem = Global.SitecoreContext.Database.GetItem(field.Value);  
        if (fieldValueItem != null)  
        {  
            list.Add(fieldValueItem);  
        }  
    }  
    return list;  
}
```

HIGHLIGHTS

X;O

Entkoppelung
von Sitecore

Testability

Test Driven
Development

- Specflows zum Definieren von Testszenarien
- Faking von:
 - Sitecore Context
 - Http Context
 - Items
 - ...
- Unit-Tests

```
Feature: Image10r15ColPresenter_Test

> Scenario Outline: Image10r15ColPresenter with different inputs
  ● Given an initialized Sitecore Context
  ● And we use these files for initialize the items:
    | FileName |
    | 2015-07-30_Image10r15ColPresenter_Items.xml |
  ● When the ImageColPresenter is called with '<datasourceItem>'
  ● Then we expect the presenter ImageColPresenter to deliver '<result>'

Scenarios:
| datasourceItem | /sitecore/content/ | test/Copy of Module Parsing Test Area/Image10r15Col/Content/Modules/Copy of Image10r15Col 1 | True |
| /sitecore/content/ | test/Copy of Module Parsing Test Area/Image10r15Col/Content/Modules/Image10r15Col | False |
```

HIGHLIGHTS

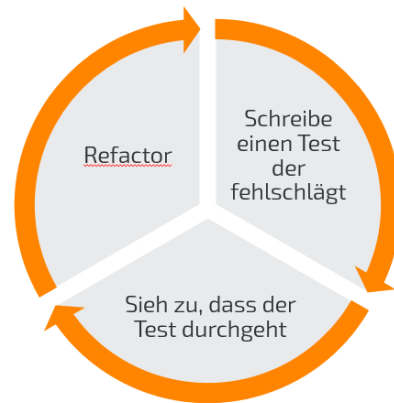
X;o

Entkoppelung
von Sitecore

Testability

Test Driven
Development

- Neuentwicklungen folgen dem TDD Ansatz
 - Testszenarien definiert in Jira (Given-When-Then) → Specflows
 - Leere Methodenstubs der Tests werden generiert
 - ... und schlagen fehl
 - (minimaler) Custom Code wird entwickelt
 - ... Tests laufen durch
 - Code wird überarbeitet
 -



Ausblick

- Ständige Weiterentwicklung des Abstraktionsframeworks
 - Vision: 100% der Sitecore.Kernel.dll 😊
- Release des Abstraktionsframeworks in die Community
 - als GitHub Projekt
 - Wird aktuell noch mit dem Kunden abgestimmt....
- Live im Einsatz seit ca. 1 Monat 😊

Vielen Dank
and
Thank You

© by ecx.io | November 15

This presentation and all of its content are the property of ecx.io. ecx.io are always at your service when it comes to implementing the ideas and solutions presented. ecx.io is free to use the content of this presentation for any other project or client. The transfer of documents to third parties, publication, reproduction, dissemination or other exploitation of the ideas and solutions is not permitted without the prior consent of ecx.io. Just give us a call.

x;o

x;o

Fragen?